Rejections under 35 U.S.C. §103

Independent Claims 12, 18, 23, 40, 47, 48, and 60 were rejected by the Examiner under 35 U.S.C. §103, and are discussed in greater detail below.

Claim 12 - Claim 12 was rejected under 35 U.S.C. §103 over U.S. 6,324,678 to Dangelo ("Dangelo '678") in view of U.S. 6,378,123 to Dupenloup ("Dupenloup '123"). Applicant has herein amended Claim 12 to include limitations relating to Applicant's invention operating at a high level of abstraction. As discussed (and shown via Applicant's demonstration) during the aforementioned interview, Applicant's invention operates at a high level of abstraction. Neither Dangelo '678 nor Dupenloup '123 teach or suggest operation at such high level of abstraction. In fact, Dangelo specifically teaches away from Applicant's invention of amended Claim 12, in that it is specifically intended to be complementary (and not supplemental) thereto. For example, the first sentence of the abstract of Dangelo '678 states the following:

> *"A method for generating structural descriptions of complex digital devices **from** high-level descriptions and specifications is disclosed."* {emphasis added}

See also, for example, Col. 2, lines 46-49 of Dangelo '678, wherein it states:

> *"A methodology for deriving a lower-level, physically implementable description, such as a RTL description of the higher level (e.g., VHDL) description, via an intermediate rule-based tool such as Prolog, is disclosed herein."* {emphasis added}

Applicant submits that the invention of Dangelo '678 is aimed only at lower levels of design abstraction. As indicated in the foregoing citation, the Dangelo '678 invention is meant to take a higher level representation (e.g., VHDL) of a design, and render a lower level description therefrom. Contrast Applicant's invention of Claim 12, which operates at high levels of abstraction. In an exemplary embodiment, Applicant's invention generates as an output the hardware description (e.g., VHDL) representation referenced by Dangelo '678. Hence, this exemplary output of Applicant's invention would be an input to the invention of Dangelo '678;

therefore, the two are complementary. Applicant's use of a high level of abstraction provides, *inter alia*, a highly intuitive, streamlined, and user-accessible design process.

Similarly, Dupenloup '123 teaches only low levels of abstraction. See, e.g., Fig. 1, block 102 of Dupenloup, and the corresponding discussion of Col. 1 generally, the former correlating to the "input" of the Dupenloup '123 methodology. Applicant's invention as claimed herein resides at the higher levels of abstraction; i.e., the output of Applicant's high level design process is used as an input to the Dupenloup '123 invention.

Similarly, U.S. 6,173,434 to Wirthlin ("Wirthlin '434) teaches manipulation of logic modules or blocks, entirely at a low level of abstraction. See, e.g., Col. 5, lines 14-19 of Wirthlin '434, which states:

> *"In the method of this invention, the development of a relocatable module also begins with the specification of the module using either a schematic diagram or text for a synthesis tool. A netlist is also produced, which can be optimized if desired. Placement and routing is also performed, as with the prior art techniques. "* {emphasis added}

Furthermore, Applicant submits that none of the other references cited by the Examiner, whether alone or in combination, teach or suggest the functionality of Claim 12 as set forth herein. U.S. 5,867,399 to Rostoker ("Rostoker '399) indirectly refers to the generation of a design in a high-level language such as VHDL (see, e.g., Col. 7, lines 53-63 of Rostocker), yet provides no teaching or details on how such design generation is performed. In fact, the aforementioned VHDL description is the input to the process described in Rostocker, analogous to Dangelo '678, Dupenloup '123, and Wirthlin '434 described above.

Applicant believes the remaining references cited by the Examiner are at best cumulative to those previously discussed herein.

Accordingly, Applicant respectfully submits that Claim 12 as amended herein can in no way be rendered obvious by the cited art, since none of the art teaches or suggest design at a high level of abstraction as specifically set forth in amended Claim 12. Hence, Applicant submits that Claim 12 defines patentable subject matter.

Claim 18 - Independent Claim 18 was similarly rejected by the Examiner over Dangelo '678 in light of Dupenloup '123. Applicant has herein amended Claim 18 to include limitations that the recited computer program is adapted to receive the input relating to a constrained set of design variables from the user. Applicant believes that none of the cited art teaches or suggests providing the user with a constrained set of design variables as in Applicant's claimed invention. Specifically, through use of such constraints, Applicant's invention of Claim 18 provides the user/designer with a highly stable basis for their design, such stability including functional predictability. See, e.g., page 4, lines 11-14, and page 16, lines 21-23 of Applicant's specification as filed. Stated simply, such constraints provide the user/designer with a "box of parts" which are known to work together and interact with the design basis (for example, the base case processor core) in predictable and stable manner. Contrast the prior art systems without such constraints, wherein the great number and degree of possible design variations generate an untenably large number of combinations, many of which are unstable, have unpredictable functionality, or are otherwise problematic from a performance aspect.

Claims 23, 40, 47, 48, and 60 - Independent Claims 23, 40, 47, 48 and 60 have been amended similar to Claim 12 discussed above; i.e., to include limitations relating to the aforementioned high level of abstraction. For reasons similar to those discussed with respect to Claim 12, Applicant respectfully submits that Claims 23, 40, 47, 48, and 60 as amended herein define patentable subject matter, and are now in condition for allowance.

Rejections under 35 U.S.C. §112

Per page 4, Par. 22 of the Office Action, the Examiner rejected Claims 32, 61, 62, and 63 under 35 U.S.C. §112 for lack of enablement. Claim 32 has been cancelled by this paper, thereby rendering such rejection moot.

With respect to the rejection of dependent Claims 61-63, Applicant respectfully traverses the Examiner's rejections as discussed below.

Claim 61 ("control logic") - Applicant refers the Examiner to, *inter alia*, page 68, line 43 of the specification as filed for support for this claim limitation. Additionally, the Examiner is referred to the computer code disclosed in Applicant's provisional application priority document, which was incorporated by reference in its entirety into the present application (see page 13, lines 7-10 of the specification as filed). Such code discloses a detailed representation of an exemplary embodiment of the referenced. pipeline control logic Applicant further submits that the generation of logic for controlling the execution stage of the instruction pipeline was well known to those of ordinary skill in the art at the time of the present invention, especially when provided the hardware description of Claim 60 from which Claim 61 depends.

Claim 62 ("instruction execution pipeline") - Applicant refers the Examiner to, *inter alia*, page 68, line 43; Fig. 3d, step 350; page 47, lines 15-16; page 65, line 45 through page 6, line 1; and page 66, lines 11-24 of the specification as filed for support for this claim limitation. Applicant submits that multi-stage instruction pipelines (including an execution stage) are notorious in the processor art; see e.g., the aforementioned prior art references. One of ordinary skill in the art at the time of the present invention would know, given Applicant's disclosure (including the aforementioned computer code), (i) how to implement a multi-stage pipeline and (ii) how to implement control logic for that pipeline based on the desired pipeline logic.

Claim 63 ("condition code choices" and "scratchpad RAM") - Applicant refers the Examiner to, *inter alia*, page 16, line 25 and page 66, lines 30-31 (condition code choices) and page 14, line 26, page 19, line 13-14, page 46, line 17, and page 47, lines 32-33 (scratchpad RAM) of the specification as filed for support for these claim limitations. See also the aforementioned computer code appendix of Applicant's incorporated provisional application. Applicant submits that one of ordinary skill in the art would know, at the date of Applicant's invention and given the present disclosure, how to implement various condition code (.cc) choices and/or a scratchpad RAM. See, e.g., United States Patent No. 5,724,563 to Hasegawa issued March 3, 1998 and entitled "Pipeline processor" (use of condition codes in a pipelined processor) and United States Patent No. 5,241,689 to Schwed, et al. issued August 31, 1993 and

entitled "Digital signal processor audio compression in an RF base station system" (describing use of scratchpad RAM in DSP).

Accordingly, based on the foregoing, Applicant respectfully submits that the Examiner's §112 rejections of Claims 61-63 are overcome.

New Claims

Applicant herein adds new Claims 75-78 for examination. Claim 76 is generally similar to prior Claim 30 (now cancelled), and Claim 77 is generally similar to prior Claim 9 (now cancelled). New Claim 78 includes, *inter alia*, limitations generally similar to those for Claim 18 as discussed above relating to the constraint of variables.

Trademark Informality

Regarding page 3, Par. 20 of the Office Action, Applicant has by this paper deleted the cited trademark symbol, thereby overcoming the Examiner's objection.

Applicant hereby specifically reserves the right to prosecute claims of different or broader scope in a continuation or divisional application.

Applicant notes that any cancellations or additions made herein are made solely for the purposes of more clearly and particularly describing and claiming the invention, and not for purposes of overcoming art or for patentability. The Examiner should infer no (i) adoption of a position with respect to patentability, (ii) change in the Applicant's position with respect to any claim or subject matter of the invention, or (iii) acquiescence in any way to any position taken by the Examiner, based on such cancellations or additions. Furthermore, the Examiner should not assume that any discussion or amendments presented above in the context of a particular claim are applicable to any other claim unless explicitly stated herein.

If the Examiner has any questions or comments which may be resolved over the telephone, he is requested to call the undersigned at (858) 675-1670.

Respectfully submitted,

GAZDZINSKI & ASSOCIATES

Dated: February 25, 2003

By: _____

Robert F. Gazdzinski
Registration No. 39,990
11440 West Bernardo Court, Suite 375
San Diego, CA 92127
Telephone No. (858) 675-1 670
Facsimile No. (858) 675-1674

METHOD AND APPARATUS FOR MANAGING THE CONFIGURATION AND FUNCTIONALITY OF A SEMICONDUCTOR DESIGN

The present application claims priority to U.S. Provisional Patent Application Serial Number 60/104,271, entitled "Method and Apparatus for Managing the Configuration and Functionality of a Semiconductor Design" filed October 14, 1998.

Copyright

1. Field of the Invention

The invention relates generally to the field of integrated circuit design. More specifically, the invention provides a method for managing the configuration, design parameters, and functionality of an integrated circuit design in which custom instructions and other elements may be arbitrarily controlled by the designer.

2. Description of Related Technology

Several types of computer aided design (CAD) tools are available to design and fabricate integrated circuits (IC). Such computer-aided or automated IC design tools can include modules or programs addressing both the synthesis and optimization processes. Synthesis is generally defined as an automatic method of converting a higher level of abstraction to a lower level of abstraction, and can include any desired combination of synthesis techniques which occur at various levels of abstraction. So-called "behavioral synthesis" is a design tool wherein the behavior (e.g. inputs, outputs, and functionality) of a desired IC are entered into a computer program to design a device that exhibits the desired behavior. Such tools permit IC designers to produce increasingly complex and capable devices, sometimes having logic gate counts in the

Appendix II is a list of the VHDL script files used in conjunction with the VHDL embodiment of the algorithm of the present invention.

It is noted that while the source code for the embodiment of the computer program set forth herein is written using AWK, a programming language commonly found on UNIX workstations, but also available on other personal computers, the program may be embodied using any of a number of different programming languages such as, for example, C^{++}. The source code directed to the exemplary AWK embodiment of the invention described herein is set forth in Applicant's aforementioned Provisional U.S. Patent Application number 60/104,271, which is incorporated herein by reference in its entirety.

Referring now to Figure 2a, one specific embodiment of the general method of Figure 2 is described. In this embodiment, certain steps of the method depicted in Figure 2 are separated into constituent parts for illustration purposes. For example, step 102 of Figure 2 is separated into an interactive ("Wizard") component and a hierarchy generator ("hiergen") component (not shown). The interactive component of step 102 provides substantially all of the direct user interaction control. Through a series of questions answered by the user, the program selects the relevant design elements necessary to realize the user's design. Portions of this information are used by the hierarchy generator to create a makefile script that is executed to build the HDL hierarchy.

Similarly, the step 104 of defining the library location for each VHDL file in Figure 2 corresponds to the steps of (i) creating a working directory 204, (ii) copying files from a master database 206, and merging the selected extension VHDL modules into placeholder files 208. The remainder of method 100 depicted in Figure 2a generally parallels that of Figure 2.

It will be appreciated by one skilled in the relevant art that there are a large number of alternative partitionings or structures of the flowchart of Figure 2, each of which results in the same or similar sets of scripts, makefiles, and other design data for a given set of input data. Further, it may be advantageous for additional data or partitionings to be selected in order to utilize commonly available tools for executing portions of the method 100. Also, the order of performance of several of the individual

Replacement Sheets for Claims

WHAT IS CLAIMED IS:

1.-11. [Cancelled]

12. An integrated circuit fabricated using the method comprising:

creating a customized description language model of an integrated circuit design by:

editing a first file specific to said design;

defining the location of at least one library file;

generating a script using said first file, said library file, and user input information; and

running said script to create said customized description language model;

generating a netlist which is descriptive of the circuitry of said integrated circuit;

compiling said netlist and said hardware description model to produce a compiled integrated circuit design;

fabricating at least one mask representing said compiled integrated circuit design; and

fabricating said integrated circuit using said at least one mask;

wherein said act of creating is performed at a high level of abstraction.

13. The integrated circuit of Claim 12, wherein the act of editing comprises selecting at least one of a plurality of input parameters associated with said design, said at least one parameter being selected from the group comprising:

(i) custom instruction sets;

(ii) cache configurations;

(iii) memory interface configurations; and

(iv) system architecture configurations.

14. The integrated circuit of Claim 12, wherein the act of generating a netlist comprises generating a list of logic devices and their interconnections.

15. The integrated circuit of Claim 12, wherein the act of fabricating said integrated circuit comprises defining physical features on a semi-conductive substrate via a lithographic process.

16. The integrated circuit of Claim 12, further comprising synthesizing said design based on said description language model.

17. The integrated circuit of Claim 13, wherein the act of editing is performed interactively with the user using a display.

18. An apparatus adapted to generate integrated circuit designs, comprising;

a processor capable of running a computer program;

a storage device operatively coupled to said processor, said storage device being capable of storing at least a portion of a computer program;

an input device, operatively coupled to said processor, capable of receiving input from a user and transmitting said input to said processor; and

a computer program resident at least in part on said storage device, said computer program adapted to receive said input relating to a constrained set of design variables from said user and perform the following acts based on said input:

editing a first file specific to said integrated circuit design;

defining the location of at least one library file;

generating a script using said first file, said library file, and user input information; and

running said script to create said description language model of said integrated circuit design.

19. The apparatus of Claim 18, wherein said description language model is a hardware description language (HDL).

20. The apparatus of Claim 18, wherein said computer program is further adapted to perform the acts comprising:

generating a second file based on said description language model for use with a simulation; and

simulating said design using said second file.

21. The apparatus of Claim 20, wherein said computer program is further adapted to perform the act comprising running synthesis scripts based on said description language model in order to synthesize said integrated circuit design.

22. The apparatus of Claim 18, wherein said processor comprises a digital microprocessor, and said storage device comprises magnetic media.

23. A system adapted for interactively generating an integrated circuit design at a high level of abstraction based on inputs received from a user, comprising:

a computer having a processor and an input device; and

a computer program capable of running on said processor, said computer program comprising:

a first algorithm having a plurality of user-selectable files, said user-selectable files comprising;

a first file comprising at least one instruction;

a second file comprising a plurality of cache configurations; and

a third file comprising a plurality of memory interface configurations;

a second algorithm capable of generating a script based on selections made by said user from said first, second, and third files and input to said computer program via said input device; and

a third algorithm capable of running said script to generate a description language model of said integrated circuit design.

24. The system of Claim 23, wherein said program is embodied in object code and stored on a storage device accessible by said processor.

25. The system of Claim 24, wherein said storage device is a rotating media magnetic storage device.

26. The system of Claim 23, said first algorithm further comprising a fourth user-selectable file, said fourth file comprising a plurality of system architectures.

27. The system of Claim 23, further comprising a fourth algorithm capable of simulating said integrated circuit design based on said description language model.

28. - 39. [Cancelled]

40. A system for generating integrated circuit designs at a high level of abstraction, comprising:

a processor;

a storage device in data communication with said processor, said storage device being capable of storing and retrieving a computer program; and

a computer program stored within said storage device and adapted to run on said processor, said computer program comprising;

a user-configurable macro-instruction having at least a first user-selectable element, said first-selectable element being selected from the group comprising;

(i) a plurality of custom instructions;

(ii) a plurality of cache configurations;

(iii) a plurality of memory interface configurations; and

(iv) a plurality of system architecture configurations;

a first algorithm capable of generating a script based on selections made by a user from said at least first user selectable element; and

a second algorithm capable of running said script to generate a description language model of an integrated circuit design

41. The system of Claim 40, wherein said computer program further comprises a second user-selectable element, said second user-selectable element allowing said user to select one of a plurality of process technology options.

42. The system of Claim 40, wherein said first user-selectable element is selected by the act of reading a pre-configured data file.

43.- 46. [Cancelled]

47. A method of generating the design of an integrated circuit [using] <u>rendered in</u> a hardware description language, said method being performed at a high level of abstraction and comprising the acts of:

selecting a process technology;

editing a first file specific to the design, said act of editing comprising selecting at least one user-configurable parameter selected from the group comprising;

(i) processor instructions;

(ii) cache configuration;

(iii) memory interface configuration; and

(iv) system architecture configuration;

defining the location of at least one library file;

generating a script using said first file and said library;

running said script to create a customized hardware description language model of the design; and

running a synthesis algorithm to synthesize a file descriptive of said design.

48. A system for generating integrated circuit designs at a high level of abstraction, comprising:

means for processing digital data;

means for data storage in data communication with said processor means, said means for data storage being capable of storing and retrieving a computer program; and

a computer program stored within said means for data storage and adapted to run on said processor means, said computer program comprising;

means for selecting a process technology;

a user-configurable macro-instruction having at least one user-selectable element, said user-selectable element being selected from the group comprising;

i) a plurality of instructions;

(ii) a plurality of cache configurations;

(iii) a plurality of memory interface configurations; and

(iv) a plurality of system architecture configurations;

means for generating a script based on said user selectable element and said process technology; and

means for running said script to generate a description language model of an integrated circuit design.

49.-59. [Cancelled]

60. A method of designing a configurable processor, the method comprising:

generating, at a high level of abstraction, a processor specification having a user-definable portion, the user-definable portion of said specification including at least one user-defined instruction having a function associated therewith; and

based on said processor specification, generating a description of a hardware implementation of said configurable processor.

61. The method of Claim 60, wherein said act of generating a description comprises generating a description including control logic necessary for the execution of said at least one user-defined instruction.

62. The method of Claim 61, wherein said act of generating a description of a hardware implementation comprises describing at least an instruction execution pipeline having a plurality of stages, said control logic including portions associated with said stages.

63. The method of Claim 60, wherein said act of generating a description comprises generating a description having at least one element selected from the group consisting of:

(i) registers; (ii) condition code choices; and (iii) scratchpad RAM.

64. The method of Claim 60, wherein said act of generating a description comprises generating a description having at least one library of multimedia extensions.

65. The method of Claim 60, further comprising simulating said configurable processor using at least said description.

66. The method of Claim 65, wherein said act of simulating comprises:

running at least one script to generate simulation data;

running at least one simulation using at least said simulation data; and

determining the adequacy of said configurable processor based at least in part on said act of running.

67. The method of Claim 60, further comprising synthesizing said configurable processor using at least said description.

68. The method of Claim 67, wherein said act of synthesizing comprises:

running at least one synthesis script to generate synthesis data;

and evaluating the adequacy of said synthesis data based at least in part on at least one design criterion.

69. The method of Claim 68, wherein said at least one design criterion comprises:

at least one specific processor performance criterion; and

at least one processor die size criterion.

70. The method of Claim 68, further comprising:

revising at least one design element when said act of evaluating indicates that said synthesis data is not adequate;

re-running said at least one synthesis script using said at least one revised design element to generate revised synthesis data;

and re-evaluating the adequacy of said revised synthesis data based at least in part on said at least one design criterion.

71. The method of Claim 70, wherein said at least one design criterion comprises at least one processor die size criterion, and said act of revising comprises revising at least one library.

72. The method of Claim 71, wherein said at least one design criterion comprises at least one processor die size criterion, and said act of revising further comprises revising at least one control file.

73. The method of Claim 70, wherein said at least one design criterion comprises processor clock speed, and said act of revising comprises revising at least one library.

74. The method of Claim 70, wherein said at least one design criterion comprises processor power consumption, and said act of revising comprises revising at least one netlist (net load).

75. A method of generating the design of an integrated circuit at a high level of abstraction using a description language, comprising the acts of:

providing an existing processor core configuration;

editing a first file specific to the design, said editing comprising selecting a constrained set of input parameters associated with said configuration, said parameters comprising:

(i) at least one custom instruction;

(ii) a cache configuration; and

(iii) a memory interface configuration;

providing at least one library file;

generating a script using said first file, said library file, and user input information;

running said script to create a customized description language model; and

synthesizing said design based on said description language model.

76. A method of generating an integrated circuit design at a high level of abstraction, comprising:

providing a user with a plurality of optional instructions, including the ability to generate a customized instruction;

selecting at least one of said plurality of optional instructions;

selecting at least one cache configuration;

defining at least one memory interface;

generating a script based on said at least one optional instruction, cache configuration, and memory interface; and

running said script to generate a hardware description language model of said integrated circuit design.

77. A description language model of an integrated circuit design generated at a high level of abstraction using the method comprising:

editing a first file specific to said integrated circuit design including selecting a plurality of input parameters associated with said design, said parameters comprising:

(i) at least one extension instruction; and

(ii) a cache configuration;

defining the location of at least one library file;

generating a script using said first file, said library file, and user input information; and

running said script to create said description language model of said integrated circuit design;

wherein said method is performed at a high level of abstraction.

78. A method of generating an extended processor design at a high level of abstraction, comprising:

providing the user with a basecase processor core configuration having a base instruction set;

providing a user with a plurality of optional instructions adaptable for use with said basecase core;

selecting at least one of said plurality of optional instructions;

selecting at least one cache configuration;

generating a script based on said at least one optional instruction, cache configuration, and basecase core; and

running said script to generate a hardware description language model of said processor design;

wherein said plurality of optional instructions and cache configurations are constrained so as to ensure the functionality of said processor design irrespective of the user's selections.